ARL-TR-7574 ● JAN 2016

US Army Research Laboratory

An Analysis Platform for Mobile Ad Hoc Network (MANET) Scenario Execution Log Data

by Jaime C Acosta and Yadira Jacquez

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-7574 ● JAN 2016

US Army Research Laboratory

An Analysis Platform for Mobile Ad Hoc Network (MANET) Scenario Execution Log Data

by Jaime C Acosta and Yadira Jacquez
Survivability/Lethality Analysis Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2016	Final	June 2015–August 2015

4. TITLE AND SUBTITLE
An Analysis Platform for Mobile Ad Hoc Network (MANET) Scenario Execution Log Data

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jaime C Acosta and Yadira Jacquez

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
Cybersecurity and Electromagnetic Protection Division
Survivability/Lethality Analysis Directorate (ATTN: RDRL-SLE-I)
White Sands Missile Range, NM 88002-5513

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-7574

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Generating models from observed binary behavior, both static and dynamic, can result in more efficient and accurate network system security analysis. Previous work in execution-based model generation has shown success for routing attacks, but requires a large amount of manual and expert-level intervention due to the lack of a graphical analysis platform. This technical report describes the ongoing work to build scenario execution data analysis platform, which is a plugin-based platform that provides a flexible and efficient mechanism for the model generation process. This platform consists of a web portal that allows analysts to set up network scenarios, view resultant data, and execute analysis algorithms.

15. SUBJECT TERMS
emulation, simulation, execution-based model generation, MANET, mobile ad hoc network, scenario execution data analysis platform, SEDAP

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jaime C Acosta
Unclassified	Unclassified	Unclassified	UU	46	19b. TELEPHONE NUMBER (Include area code) (575) 678-8115

Contents

List of Figures

List of Tables

Acknowledgements

This work was completed as part of the US Army Research Laboratory, Survivability/Lethality Analysis Directorate and the University of Texas at El Paso (UTEP) open campus initiative. From UTEP, we would like to thank Dr Salamah Salamah and Adrian Garcia for their help with this work.

1. Introduction

1.1 Problem

Testing the security posture of network systems is a critical step in the Army acquisition lifecycle. While field tests are necessary because they provide the highest level of accuracy, they are very costly, limited in duration, and are difficult to coordinate. For this reason, system models are developed to enable experimentation with emulation and simulation in laboratory environments—aiding both in field test preparation and in providing supplementary post-mission analysis from field test data. Both simulation and emulation have advantages; simulation supports faster-than-real-time execution and complete control over the environment while emulation tools provide results that are closer to the ground truth by supporting real binary execution (i.e., they are capable of executing binaries that run on actual systems).

1.2 Approach

The Cybersecurity and Electromagnetic Protection Division (CEPD) branch of the US Army Research Laboratory (ARL) has developed a unique analysis approach that exploits the advantages of both emulation and simulation. This approach uses the results from several emulation executions with realistic scenarios to automatically develop accurate models that can be used in, for example, simulators. These models may be decision trees or complex algorithms and formulas. This approach differs from traditional workflows where models are developed and tested before or alongside the system development. Some issues with the traditional approach include lack of synchronization between the actual system and the models due to changes in requirements, manual development of highly accurate models may be too expensive, and the models may not be made available to analysts (either for legal reasons or due to nonexistent models). The novel approach starts instead with the end-product (i.e., the executable code). By using the end-product it is possible to extract not only models for the intended behavior of the systems, but also incidental models such as resilience to unanticipated adversarial attacks.

An instance where this work has shown success is in the field of impact analysis of attacks in mobile ad hoc networks (MANETs). First, executables that are commonly used in MANETs such as routing protocol implementations and Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) traffic generators along with both in-house and publicly available attacks on the International Organization of Standardization (ISO) network Layers 3–7 were

identified. Next, to develop models for Army strategic and tactical technologies, state-of-the-art emulators (e.g., the common open research emulator [CORE] and the mobile ad hoc network emulator [MANE] were used to develop and execute several scenarios. Each scenario consisted of various configuration parameters; during each scenario several data were logged. Finally, these data were then analyzed at the mid-grain level and models exhibiting high precision and recall measures were developed.

1.3 Objectives

While this work has exhibited success, there is much room for improvement. A critical factor in this approach is the analysis of the data collected during scenario execution. Scenarios consist of node topologies, traffic flows, routing protocols, and attacks on the network. Figure 1 shows the experimentation workflow.

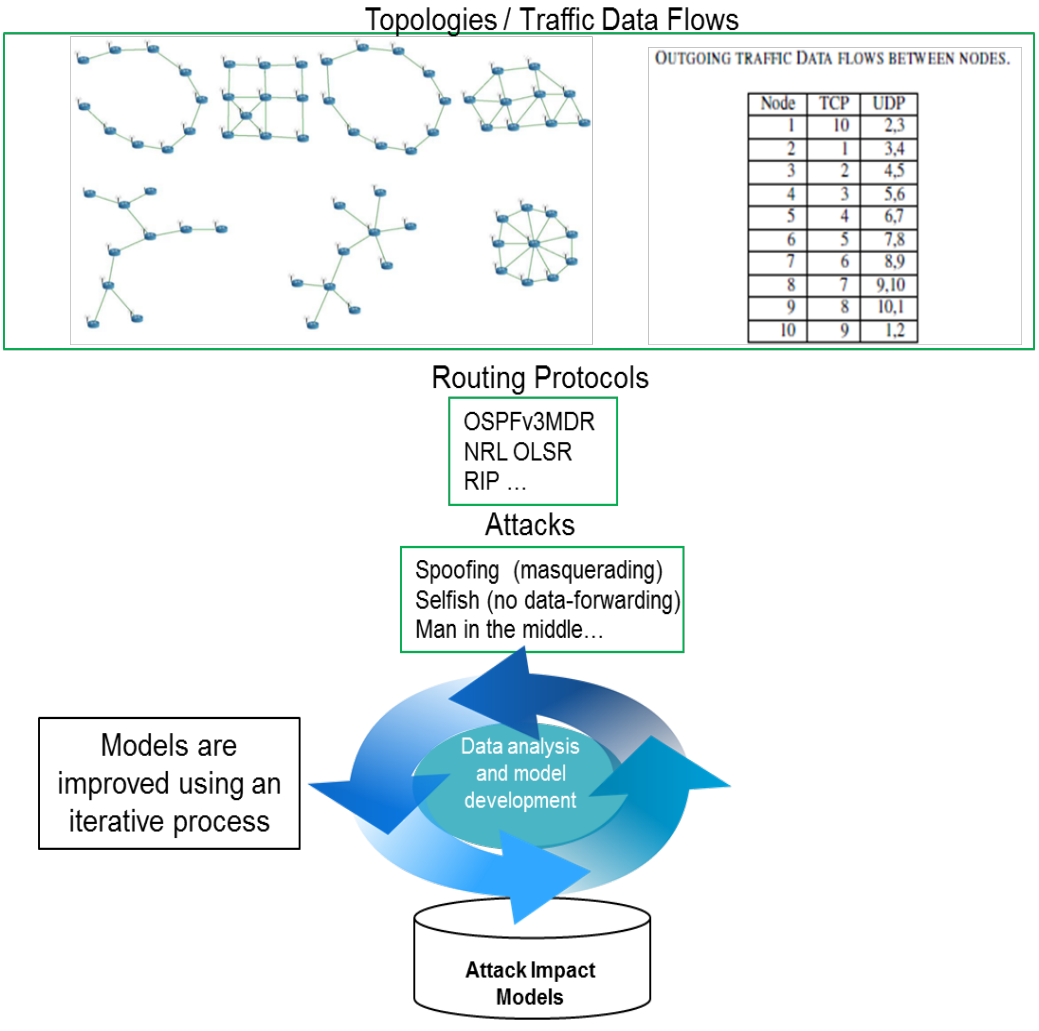

Fig. 1 Experimentation workflow

The experimentation workflow consists of running several network scenarios. An example of a single scenario is the following: chain topology (the topology in the top left [refer to Fig.1]), US Naval Research Laboratory Optimized Link-State Routing (NRL OLSR) Protocol used for routing, Node 1 issues a selfish attack. Scenarios are automatically generated and executed with python and bash scripts. The data generated from these scenarios are converted, with scripts, to Waikato Environment for Knowledge Analysis (WEKA) format. WEKA is a machine learning software suite and other statistical analysis toolsets. These analysis toolsets are used to generate models. The issue is that the quality of the models is dependent on the data collected during the scenario execution, the scenario configurations, and the algorithms used in the statistical tools. Currently there is no way to manage the data; sifting through the data is very time consuming and many times inhibits the ability to quickly identify attributes that may improve the accuracy of the models.

In this report we describe our initial design and implementation of the scenario execution data analysis platform (SEDAP) that aims at providing the following functionality:

- A data store that keeps all log data in a structured format and associate the logs with a scenario, analyst (whoever executed the scenario) and maintain information about duplicate executions.
- A graphical frontend that an analyst can use to identify data of interest, run and queue scenarios, compare data, and preview portions of data (i.e., a quick look feature).
- The capability to handle flexible data, that is, the data collected during a scenario may change by adding attributes or modifying the format of the data.

A simple scenario of how this software may help an analyst follows. After running WEKA, the attack impact model quality is worse than expected. An analyst proceeds to further investigate the data. There exist 3 scenarios that are very similar and should yield similar results. An analyst could use this software to compare/contrast/rerun scenarios, and so forth.

2. Analysis of the Experimental Workflow

To design SEDAP we first conducted an informal investigation of the current data structure along with the collection and model generation process. A virtual machine contained the experimental setup along with all of the scripts and external tools that were required. The following are the steps that we followed to recreate the attack

models (i.e., these are the detailed instructions for executing the workflow shown in Fig. 1).

2.1 Collecting Data with CORE

CORE is an open source tool that was originally developed for experimenting with network technologies, its primary focus is on performance. The MANET project at ARL, Survivability/Lethality Analysis Directorate (SLAD) is using CORE to develop efficient ways to analyze the effects of attacks (primarily at the network layer) on systems. More information about CORE can be found at their website (Official Navy Website).

We followed these steps to download, install, and run CORE.

1) Download the experimentation virtual machine (this includes CORE and other required packages) from the following Uniform Resource Locator (URL):

   https://drive.google.com/open?id=0B4bNFE1fnY-GR3FUWUh5MGNkUWc&authuser=0.

2) Import the virtual machine into VirtualBox.

3) Start the virtual machine and login with valid credentials.

4) To become more familiar with CORE, start the process by opening a terminal window and typing: core-gui.

5) To run the experimentation workflow, open a terminal and type the following: cd /root/IntelAttacker/./generateScenarios.sh.

The screen should now resemble the screenshot shown in Fig. 2.

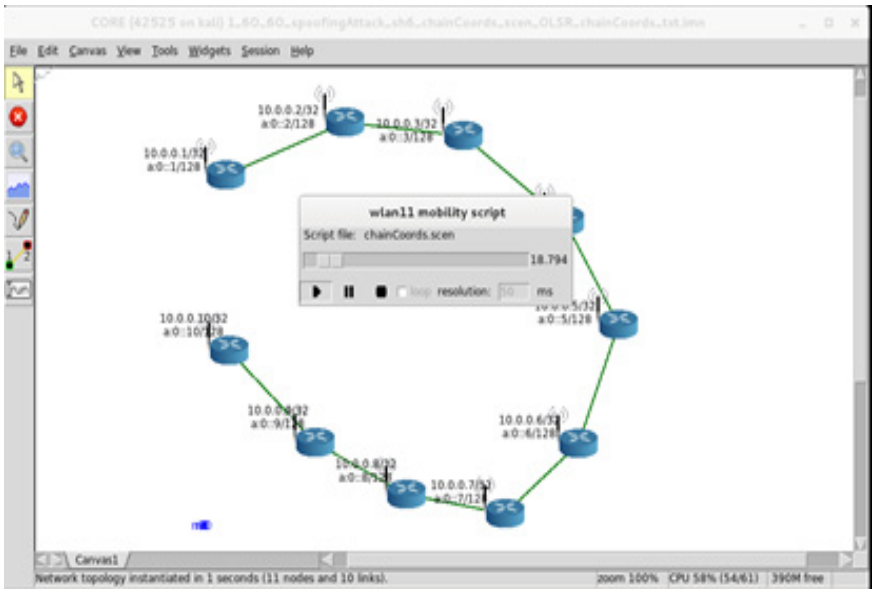

Fig. 2 CORE scenario screenshot

Each scenario has 3 phases: beforeAttack, duringAttack, and afterAttack. These phases are each 60 s in length, (i.e., at time 60, one of the nodes starts an attack; at time 120 the attack ceases). When the time indicator reaches 180.0, this scenario completes and CORE restarts with a new scenario. Data from each scenario resides in a folder in the /root/ directory. The entire experiment takes several hours to complete and consists of several thousands of scenario executions. Figure 3 shows the directory structure of the data resulting from a scenario execution.

```
1  <scenario name>
2      <attacker.capture>
3      <nonattacker.mgencapture>
```

Fig. 3 Log directory structure

2.1.1 CORE Data File

The execution of a scenario in CORE generates 2 types of data files: .capture and .mgencapture. An example of the data in these files is described in the following subsections.

2.1.2 Capture File (Raw Data)

The raw data .capture file consists of one data entry per line. Each data entry is made up of the data attributes shown below separated by a semicolon (;).

A sample data entry follows:

11;{('10.0.0.2_224.0.0.57', 'eth:ip:udp:olsr'): ('1', '*')}; {'224.0.0.0': ('0', '0.0.0.0'), '10.0.0.10': ('9', '10.0.0.2'), … '10.0.0.2': ('1', '0.0.0.0')}; none

2.1.3 Mgencapture File (Raw Data)

The raw data .mgencapture file consists of one data entry per line. Each data entry is made of the data attributes shown below separated by a semicolon (;).

A sample data entry follows:

5; {('10.0.0.5_10.0.0.2', 'TCP'): (34.3137370000004, 0, 48, 51,'3'), … , ('10.0.0.10_10.0.0.2', 'UDP'): (46.422825000000365, 0,0,50,'8')}; 464.311602; 1; 321; 627; none; {"224.0.0.0': ('0', '0.0.0.0'), '10.0.0.10': ('8', '10.0.0.3'), ... , '10.0.0.3': ('1', '0.0.0.0')}

2.2 Using WEKA to Generate Model

WEKA is a data mining platform that consists of a suite of algorithms that can be used to generate classifiers.

We followed the following steps to generate the models from the data collected in Section 2.1.

Download and install WEKA from here:

1) Download and install WEKA from:
   http://www.cs.waikato.ac.nz/ml/WEKA/downloading.html
   (A separate WEKA tutorial for beginners is found here:
   http://www.ibm.com/developerworks/library/os-WEKA1/.

2) Start WEKA.

3) The script /root/IntelAttacker/runAllToArff.sh converts the scenario data into a format that can be read by WEKA. This has already been done for you; download the resulting WEKA-readable file from:
   https://drive.google.com/file/d/0B4bNFE1fnY-GVmxscWVydFJTRTA/view?usp=sharing.

4) Decompress the file and open it with WEKA.

5) Remove the following attributes: 1, 2, 3, 13–20, 25, 26.

The screen should now resemble the screenshot in Fig. 4.

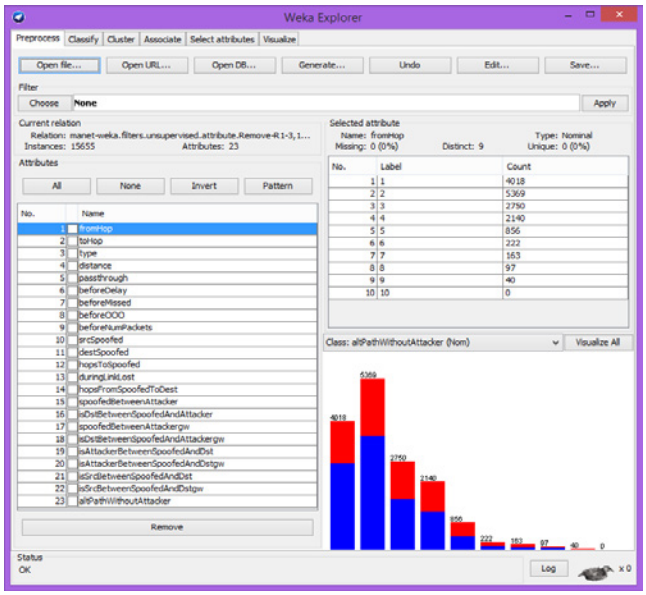

Fig. 4 WEKA preprocess window

6) Click the **Classify** tab.
7) Click the **Choose** button.
8) Find and select the **REPTree** classifier.
9) In the drop-down menu, select **(Nom) duringLinkLost**.
10) Click the **Start** button.

The screen should now resemble the screenshot in Fig. 5.

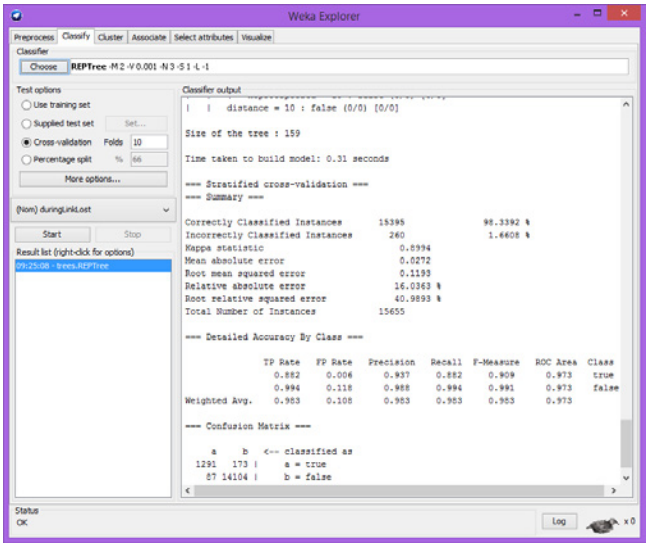

Fig. 5 WEKA classify window

2.2.1 Attribute-Relation File Format (Arff) File

The Arff file is a composite data file generated from the raw data contained in the .capture and .mgencapture files. It consists of one data entry per line. Each data entry is made up of the data attributes shown below separated by a comma (,).

A sample data entry follows:

/root/1_60_60_spoofingAttack_sh_1_chainCoords_scen_OLSR_chainCoords_txt,
1,10.0.0.2_10.0.0.9,1,8,UDP,7,false,37.19573757,0.133333333,0.133333333,43.3
3333333,-5.468208683,0.133333333,-6.666666667,-
5.7378654,0.033333333,0.033333333,-
6.633333333,false,false,0,false,false,spoof_10.0.0.1, -1,
true,false,false,true,true,true,true,true,true,true

2.3 Observations

After stepping through the experimental workflow and analyzing the structure of log files, we identified the following technologies that were essential for building SEDAP.

- A database to store the parsed data in the .mgencapture, .capture, and Arff files.
- A web interface that can be used to view the data in an organized and searchable way.
- Backend web services that can execute CORE, other processing scripts, and connect the web interface with the database.

The following sections describe the design and implementation details of SEDAP.

3. SEDAP

The SEDAP is currently accessible through a web interface and provides 3 main functions: Data Parsing, Conflict Detection, and Scenario Execution Automation.

3.1 Data Parsing

The Data Parsing function consists of a tool that reads the 3 types of data files generated by the CORE tool upon scenario execution and from the Arff file, extracts the information, and places it into the database. The tool takes a directory path as a parameter and parses all the files within that directory and subdirectories.

The data from the .capture file is stored in the Capture table of the database; the data from the .mgencapture file is stored in the MgenCapture table of the database; and the data from the Arff file is stored in the Arff table in the database. Additionally, data from the .capture file and the .mgencapture file are stored in the CaptureFlows table and the MgenCaptureFlows table, respectively. Finally, partial data from both the .capture and .mgencapture files are stored in the Routes table.

See Section 5 for information on specific data fields on each of the database tables previously mentioned.

3.2 Conflict Detection

The Conflict Detection function consists of a tool that analyses the Arff table data in the database and determines if there are any "conflicts" in each of the scenarios stored in this table.

Figure 6 shows the pseudo code for the algorithm used to identify a conflict.

```
1  For each line in <.arff file>
2     Key <- all attributes except "duringLinkLost"
3     Value <- "duringLinkLost" attribute
4     If (HashTable contains Key)
5        If (HashTable[Key] != Value)
6           Tag data conflict
7        Else (HashTable[Key] = Value)
```

Fig. 6 Pseudo code for the conflict detection algorithm

The attributes listed below make up what is referred to as a "key".

***@attribute fromHop {1,2,3,4,5,6,7,8,9,10}*
***@attribute toHop {1,2,3,4,5,6,7,8,9,10}*
***@attribute type {TCP,UDP}*
***@attribute distance {1,2,3,4,5,6,7,8,9,10}*
***@attribute passthrough {true, false}*
***@attribute srcSpoofed {true, false}*
***@attribute destSpoofed {true, false}*
***@attribute hopsToSpoofed {0,1,2,3,4,5,6,7,8,9,10}*
***@attribute attackName {forwarding, down, spoof_10.0.0.1, spoof_10.0.0.2, spoof_10.0.0.3, spoof_10.0.0.4, spoof_10.0.0.5, spoof_10.0.0.6, spoof_10.0.0.7, spoof_10.0.0.8, spoof_10.0.0.9, spoof_10.0.0.10}*
***@attribute hopsFromSpoofedToDest numeric*
***@attribute attackerCloserToDestThanSpoofed {true, false}*
***@attribute spoofedBetweenAttacker {true, false}*

***@attribute isDstBetweenSpoofedAndAttacker {true, false}*
***@attribute spoofedBetweenAttackergw {true, false}*
***@attribute isDstBetweenSpoofedAndAttackergw {true, false}*
***@attribute isAttackerBetweenSpoofedAndDst {true, false}*
***@attribute isAttackerBetweenSpoofedAndDstgw {true, false}*
***@attribute isSrcBetweenSpoofedAndDst {true, false}*
***@attribute isSrcBetweenSpoofedAndDstgw {true, false}*
***@attribute altPathWithoutAttacker {true, false}*

For every row in the Arff table, the following attribute is referred to as the "value"*:*

****@attribute duringLinkLost {true, false}*

A conflict is said to have occurred when 2 or more rows share the same key but have a different value.

The Conflict Detection tool examines the entire Arff table in the database and looks for such conflicts. When one is found, all the rows that share the same key are copied to the Conflict table in the database. Every time the Conflict Detection function is executed, the Conflict table is cleared (all existing conflicts are dropped from the table) and the entire Arff table is analyzed for conflicts populating the Conflict table, if any are found.

3.3 Scenario Automation

The Scenario Automation function consists of a tool that automates the execution of a scenario in CORE, waits for the data files to be generated by CORE, and finally parses these files into the database, as described in Section 3.1.

Execution begins by selecting the attributes (in the frontend) needed to generate a new scenario in CORE: topology, protocol, attack, and attack node. The tool receives these parameters through a web service and executes a bash script, "sedap.sh". This script serves 2 purposes, first, it sets the working directory in Linux to /root/IntelAttacker. Note that the CORE application will not start unless it is called from this specific directory in the Virtual Machine directory structure. Second, the script calls the bash script (generateScenariosSedap.sh) that actually starts CORE and passes it the 4 parameters referenced above to execute a scenario.

After CORE has been invoked, the tool waits 190 s for CORE to complete executing the new scenario (CORE takes 180 s to complete). Once the wait is over, the Data Parser tool is invoked by passing the directory path in which the CORE data files were generated.

4. Technologies

The following technologies were used in the development of the SEDAP system. Specific version numbers, when noted, should be followed as compatibility issues were encountered between different versions of these technologies.

4.1 Backend Technologies

- Java 1.8
- my-sql-connector-java-5.0.8.jar
- Tomcat
- VirtualBox
- Kali MANET Virtual Machine

4.2 Frontend Technologies

- LAMPP

4.3 Database

- MySQL Server

5. Database

The SEDAP database settings and structure are described in this section.

5.1 Database Schema

Figure 7 shows the database schema. The schema is presented in an informal notation, and a key is included to describe what each element represents. Elements include data entities, which hold information that need to be stored, and relationships (shown as arrows) between the data entities. The relationships, labeled with attribute names, represent how rows from 2 tables are related to each other; rows from one table are grouped with rows from another table if they have the same attribute value(s).

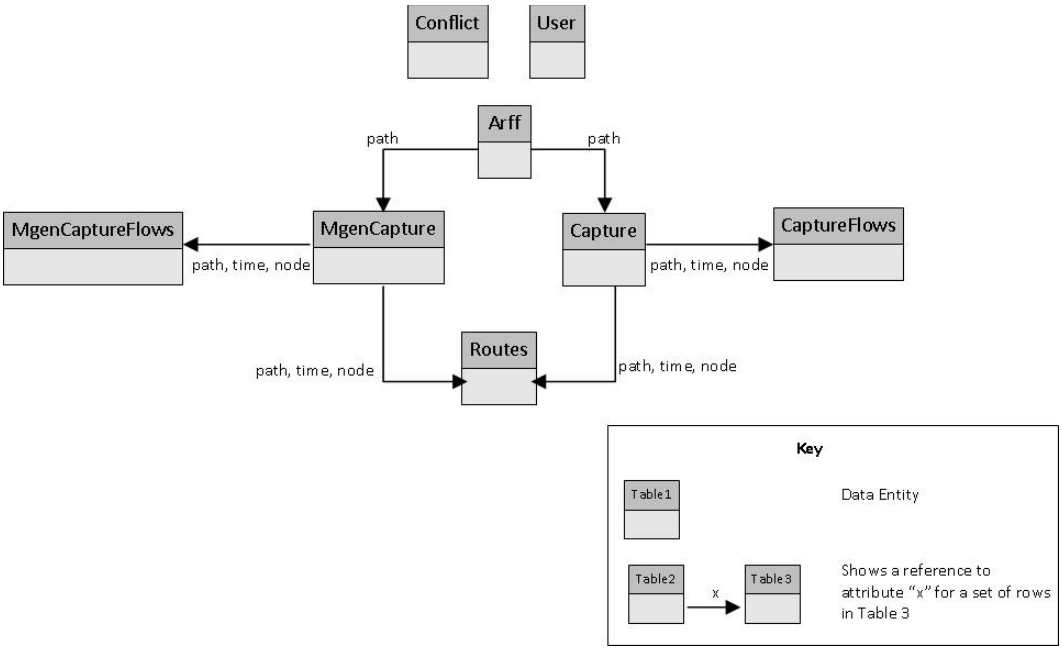

Fig. 7 SEDAP object diagram

5.2 Database Table Catalog

The database consists of 8 tables: Arff, Capture, CaptureFlows, Conflict, MgenCapture, MgenCaptureFlows, Routes, and User.

5.2.1 Arff

The Arff table stores data from the Arff file. Table 1 lists the attributes and the implementation details of the Arff table.

Table 1 Arff table fields

Attribute	Type
path	Varchar(255)
attackNodeNum	Int
description	Varchar(50)
fromHop	Int
toHop	Int
type	Varchar(50)
distance	Int
passthrough	Boolean
beforeDelay	Double
beforeMissed	Double
beforeOOO	Double
beforeNumPackets	Double
duringDelay	Double
duringMIssed	Double
duringOOO	Double
duringNumPackets	Double
afterDelay	Double
afterMissed	Double
afterOOO	Double
afterNumPackets	Double
srcSpoofed	Boolean
destSpoofed	Boolean
hopsToSpoofed	Int
duringLinkLost	Boolean
afterLinkLost	Boolean

Table 1 Arff table fields (continued)

Attribute	Type
attackName	Varchar(50)
spoofedBetweenAttacker	Boolean
isDstBetweenSpoofedAndAttacker	Boolean
spoofedBetweenAttackergw	Boolean
isDstBetweenSpoofedAndAttackergw	Boolean
isAttackerBetweenSpoofedAndDst	Boolean
isAttackerBetweenSpoofedAndDstgw	Boolean
isSrcBetweenSpoofedAndDst	Boolean
isSrcBetweenSpoofedAndDstgw	Boolean
altPathWithoutAttacker	Boolean

5.2.2 Capture

The Capture table stores partial data from the .capture file. Table 2 lists the attributes and the implementation details of the Capture table.

Table 2 Capture table fields

Attribute	Type
path	Varchar(255)
time	Int
node	Int
attackRunning	Varchar(50)

Note: Each data entry from the .capture file includes more attributes than is described in this table. The additional data is stored in 2 separate tables: CaptureFlows and Routes.

5.2.3 CaptureFlows

The CaptureFlows table stores partial data from the .capture file. Table 3 describes the attributes and the implementation details of the CaptureFlows table.

Table 3 Captureflows table fields

Attribute	Type
path	Varchar(255)
time	Int
node	Int
flow	Varchar(50)
proto	Varchar(80)
hopsToSrc	Varchar(10)
hopsToDst	Varchar(10)

Note: Each data entry from the .capture file includes more attributes than is described in this table. The additional data is stored in 2 separate tables: Capture and Routes.

5.2.4 Conflict

The Conflict table stores the data entries from the Arff files that are in conflict. Because this table stores .Arff data entries, the Conflict table is a replica of the Arff table.

5.2.5 MgenCapture

The MgenCapture table stores partial data from the .mgencapture file. Table 4 lists the data and the implementation details of the MgenCapture table.

Table 4 MgenCapture table fields

Attribute	Type
path	Varchar(255)
time	Int
node	Int
totalDelay	Double
totalMissedPackets	Double
totalOOO	Double
totalNumPackets	Double
attackRunning	Varchar(50)

Note: Each data entry from the .mgencapture file includes more attributes than is described in this table. The additional data is stored in 2 separate tables: MgenCaptureFlows and Routes.

5.2.6 MgenCaptureFlows

The MgenCaptureFlows table stores partial data from the .mgencapture file. Table 5 lists the attributes and the implementation details of the MgenCapture table.

Table 5 MgenCaptureFlows table fields

Attribute	Type
path	Varchar(255)
time	Int
node	Int
flow	Varchar(50)
proto	Varchar(50)
delay	Double
missedPackets	Double
ooo	Double
numPackets	Double
dist	Varchar(3)

Note: Each data entry from the .mgencapture file includes more attributes than is described in this table. The additional data is stored in 2 separate tables: MgenCapture and Routes.

5.2.7 Routes

The Routes table stores partial data from both the .capture and .mgencapture files. Table 6 lists the data and the implementation details of the Routes table.

Table 6 Routes table fields

Attribute	Type
path	Varchar(255)
time	Int
node	Int
route	Varchar(5000)

5.2.8 User

The User table is intended to store the user data for each user. Table 7 lists the data and the implementation details of the User table.

Table 7 User table fields

Attribute	Type
Uusername	Varchar(30)
Ufirst_name	Varchar(30)
Ulast_name	Varchar(30)
Upassword	Varchar(100)
Usalt	Varchar(60)

Note: The SEDAP currently does not support user profiles. The table was created to support future requirements.

5.3 Initial Database Load

Upon installing a new database server and creating the database schema previously described, SEDAP provides the developers the ability to perform an initial data load. This can be achieved by running the Data Parsing functionality as a standalone tool. The LogParser tool can be executed from its “main()” method by calling the “parseAllFilesInDirectory(directory)” method, where “directory” is the root directory that contains the initial set of CORE data files to be parsed and loaded into the database.

6 Backend

The backend directory structure (on the Virtual Machine) and architecture are described in the following sections.

6.1 File Structure

The Virtual Machine that we used as our development environment contains a specific directory structure that allows for the correct execution of the backend tools.

CORE generates the data files and stores them in the /root/ directory. These data files must remain in this directory for the SEDAP application to function correctly.

Additionally, 2 bash scripts described in Section 3.3, sedap.sh and generateScenarioSedap.sh, must be located in the /root/IntelAttacker/ directory for the SEDAP application to function correctly.

The .WAR file, which contains all the backend java functionality including the web services, should be placed in the webapps directory inside the Tomcat installation directory.

6.2 Architecture

Figure 8 shows the SEDAP backend system modules. When a module is connected to another by a dashed arrow, the first module uses the latter. The correct functionality of a module that uses another module depends on the correct implementation of the second. The diagram is presented in an informal notation, and a key is included to describe what each element represents.

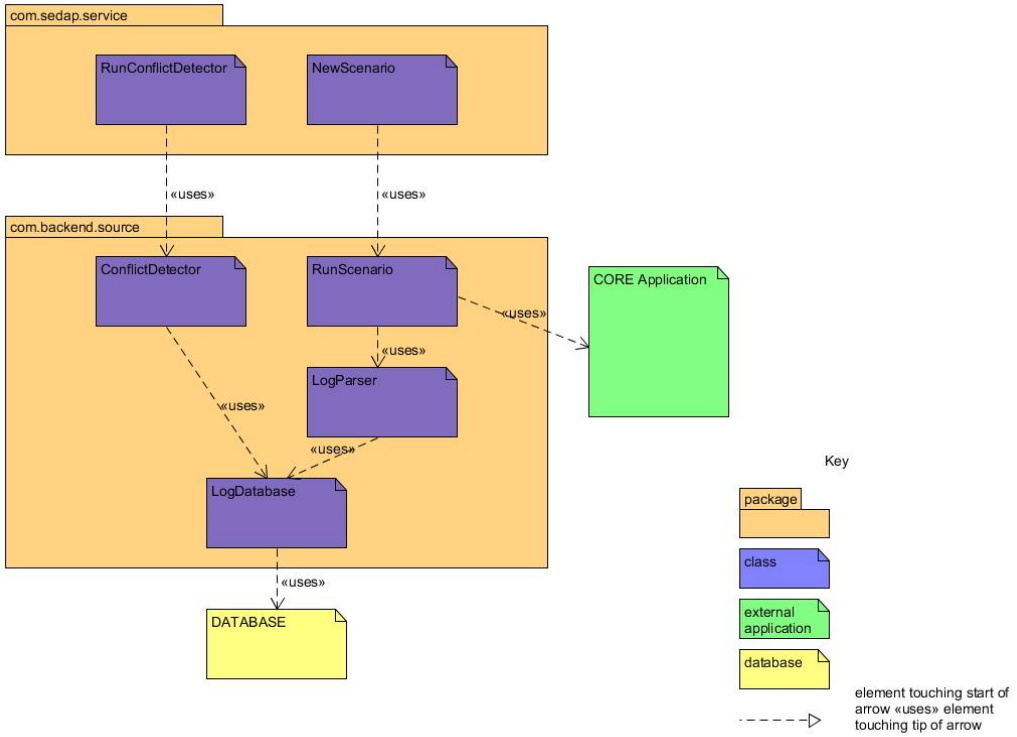

Fig. 8 SEDAP backend module diagram

6.3 Module Catalog

The modules, represented as class elements in Fig. 8, are described in this section.

6.3.1 LogParser.java

The LogParser class is a static class/stand-alone tool that takes a directory path as a parameter and recursively parses all the files of type .capture, .mgencapture, and .arff in the root directory of the path and all its subdirectories. The parsed data is then stored in the database.

LogParser <<uses>> LogDatabase

6.3.2 ConflictDetector.java

The ConflictDetector class is a static class/stand-alone tool that analyses the data in the Arff table of the database and stores detected conflicts in the Conflict table of the database.

ConflictDetector <<uses>> LogDatabase

6.3.3 LogDatabase.java

The LogDatabase class is used as an interface to the SEDAP database. The class provides methods for database connection and query execution.

LogDatabase <<uses>> DATABASE

6.3.4 RunScenario.java

The RunScenario class is a static class/stand-alone tool that takes 4 attributes as parameters, triggers the CORE application to execute a new scenario using the 4 attributes, and finally executes the LogParser by passing the root directory of the newly generated CORE scenario.

RunScenario <<uses>> LogParser
<<uses>> CORE (external application)

6.3.5 NewScenario.java

The NewScenario REST service is used as the interface between the frontend web application and the RunScenario tool.

NewScenario <<uses>> RunScenario

6.3.6 RunConflictDetector.java

The RunConflictDetector REST service is used as the interface between the frontend web application and the ConflictDetector tool.

RunConflictDetector <<uses>> ConflictDetector

7. Frontend

The frontend directory structure (on the Virtual Machine) and architecture are described in the following sections.

7.1 File Structure

The SEDAP web interface pages are stored in the XAMPP root directory, htdocs. On our Kali Linux Virtual Machine, the full path for the root directory is /opt/lampp/htdocs/xampp/. The web pages are packaged within a folder named FINAL. To request the web application from the localhost server, the URL localhost must be followed by /FINAL.

7.2 Architecture

Figure 9 shows the interaction between the SEDAP web interface and the external entities—the database and the REST services. Only a subset of the pages from the web application are shown in this diagram. The subset includes only the pages that are interacting with the external entities. The diagram is presented in an informal notation, and a key is included to describe what each element represents.

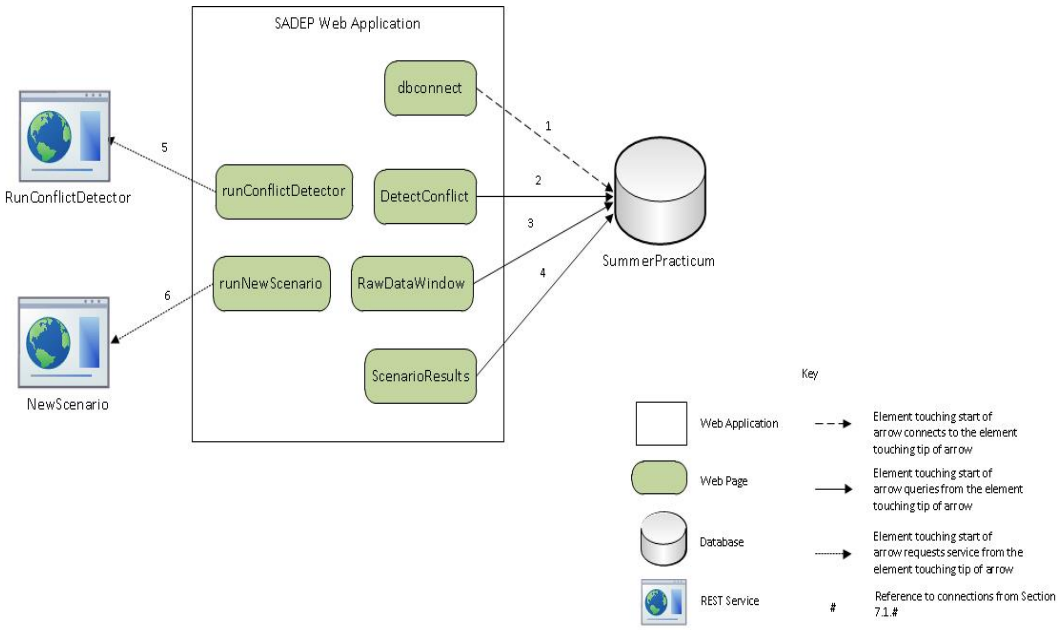

Fig. 9 SEDAP web service diagram

7.3 Connection Catalog

The SEDAP web interface interacts with the external entities via connections. These connections are in the form of mysqli queries and REST requests, for interfacing with the database and with the REST services, respectively. These connections are described in the following sections.

7.3.1 dbconnect

The following query is used to establish a connection with the database:

new mysqli(<server>, <username>, <password>, <database name>);

7.3.2 DetectConflict

1) The following query is used to select all the distinct keys from the Conflict table. The distinct keys are used to group the rows that are in conflict.

   mysqli_query($connection, "SELECT DISTINCT fromHop, toHop, type, distance, passthrough, srcSpoofed, destSpoofed, hopsToSpoofed, attackName, hopsFromSpoofedToDest, attackerCloserToDestThanSpoofed, spoofedBetweenAttacker, isDstBetweenSpoofedAndAttacker, spoofedBetweenAttackergw, isDstBetweenSpoofedAndAttackergw, isAttackerBetweenSpoofedAndDst, isAttackerBetweenSpoofedAndDstgw, isSrcBetweenSpoofedAndDst, isSrcBetweenSpoofedAndDstgw, altPathWithoutAttacker FROM Conflict");

2) The following query is used to count the number of times the value, duringLinkLost, evaluates to true for the group of rows from the Conflict table that are in conflict. This value is displayed on the DetectConflict page.

mysqli_query($connection, "SELECT COUNT(duringLinkLost) AS trueCount FROM Conflict WHERE fromHop ='".$path['fromHop'] ."' AND toHop ='".$path['toHop'] ."' AND type='".$path['type'] ."' AND distance='".$path['distance'] ."' AND passthrough='".$path['passthrough'] ."' AND srcSpoofed='".$path['srcSpoofed'] ."' AND destSpoofed='".$path['destSpoofed'] ."' AND hopsToSpoofed='".$path['hopsToSpoofed'] ."' AND attackName='".$path['attackName'] ."' AND hopsFromSpoofedToDest='".$path['hopsFromSpoofedToDest'] ."' AND attackerCloserToDestThanSpoofed='".$path['attackerCloserToDestThanSpoofed'] ."' AND spoofedBetweenAttacker ='".$path['spoofedBetweenAttacker'] ."' AND isDstBetweenSpoofedAndAttacker ='".$path['isDstBetweenSpoofedAndAttacker'] ."' AND spoofedBetweenAttackergw ='".$path['spoofedBetweenAttackergw'] ."' AND isDstBetweenSpoofedAndAttackergw ='".$path['isDstBetweenSpoofedAndAttackergw'] ."' AND isAttackerBetweenSpoofedAndDst ='".$path['isAttackerBetweenSpoofedAndDst'] ."' AND isAttackerBetweenSpoofedAndDstgw ='".$path['isAttackerBetweenSpoofedAndDstgw'] ."' AND isSrcBetweenSpoofedAndDst ='".$path['isSrcBetweenSpoofedAndDst'] ."' AND isSrcBetweenSpoofedAndDstgw ='".$path['isSrcBetweenSpoofedAndDstgw'] ."' AND altPathWithoutAttacker ='".$path['altPathWithoutAttacker'] ."' AND duringLinkLost = 'true'");

3) The following query is used to count the number of times the value, duringLinkLost, evaluates to false for the group of rows from the Conflict table that are in conflict. This value is displayed on the DetectConflict page.

mysqli_query($connection, "SELECT COUNT(duringLinkLost) AS falseCount FROM Conflict WHERE fromHop ='".$path['fromHop'] ."' AND toHop ='".$path['toHop'] ."' AND type='".$path['type'] ."' AND distance='".$path['distance'] ."' AND passthrough='".$path['passthrough'] ."' AND srcSpoofed='".$path['srcSpoofed'] ."' AND destSpoofed='".$path['destSpoofed'] ."' AND hopsToSpoofed='".$path['hopsToSpoofed'] ."' AND attackName='".$path['attackName'] ."' AND hopsFromSpoofedToDest='".$path['hopsFromSpoofedToDest'] ."' AND attackerCloserToDestThanSpoofed='".$path['attackerCloserToDestThanS

poofed'] ."' AND spoofedBetweenAttacker ='".$path['spoofedBetweenAttacker'] ."' AND isDstBetweenSpoofedAndAttacker ='".$path['isDstBetweenSpoofedAndAttacker'] ."' AND spoofedBetweenAttackergw ='".$path['spoofedBetweenAttackergw'] ."' AND isDstBetweenSpoofedAndAttackergw ='".$path['isDstBetweenSpoofedAndAttackergw'] ."' AND isAttackerBetweenSpoofedAndDst ='".$path['isAttackerBetweenSpoofedAndDst'] ."' AND isAttackerBetweenSpoofedAndDstgw ='".$path['isAttackerBetweenSpoofedAndDstgw'] ."' AND isSrcBetweenSpoofedAndDst ='".$path['isSrcBetweenSpoofedAndDst'] ."' AND isSrcBetweenSpoofedAndDstgw ='".$path['isSrcBetweenSpoofedAndDstgw'] ."' AND altPathWithoutAttacker ='".$path['altPathWithoutAttacker'] ."' AND duringLinkLost = false");

4) The following query uses the key to select the group of rows from the Conflict table that are in conflict. The group is displayed on the DetectConflict page.

mysqli_query($connection, "SSELECT * FROM Conflict WHERE fromHop ='".$path['fromHop'] ."' AND toHop ='".$path['toHop'] ."' AND type='".$path['type'] ."' AND distance='".$path['distance'] ."' AND passthrough='".$path['passthrough'] ."' AND srcSpoofed='".$path['srcSpoofed'] ."' AND destSpoofed='".$path['destSpoofed'] ."' AND hopsToSpoofed='".$path['hopsToSpoofed'] ."' AND attackName='".$path['attackName'] ."' AND hopsFromSpoofedToDest='".$path['hopsFromSpoofedToDest'] ."' AND attackerCloserToDestThanSpoofed='".$path['attackerCloserToDestThanSpoofed'] ."' AND spoofedBetweenAttacker ='".$path['spoofedBetweenAttacker'] ."' AND isDstBetweenSpoofedAndAttacker ='".$path['isDstBetweenSpoofedAndAttacker'] ."' AND spoofedBetweenAttackergw ='".$path['spoofedBetweenAttackergw'] ."' AND isDstBetweenSpoofedAndAttackergw ='".$path['isDstBetweenSpoofedAndAttackergw'] ."' AND isAttackerBetweenSpoofedAndDst ='".$path['isAttackerBetweenSpoofedAndDst'] ."' AND isAttackerBetweenSpoofedAndDstgw ='".$path['isAttackerBetweenSpoofedAndDstgw'] ."' AND isSrcBetweenSpoofedAndDst ='".$path['isSrcBetweenSpoofedAndDst'] ."' AND isSrcBetweenSpoofedAndDstgw ='".$path['isSrcBetweenSpoofedAndDstgw'] ."' AND altPathWithoutAttacker ='".$path['altPathWithoutAttacker'] ."'");

7.3.3 RawDataWindow

1) The following query is used to select the raw data from the Capture table that is associated with the node of interest.

   mysqli_query($connection, "SELECT * FROM Capture WHERE path='".$pathname."/".$nodeFile."');

2) The following query is used to select the raw data from the MgenCapture table that is associated with the node of interest.

   mysqli_query($connection, "SELECT * FROM MgenCapture WHERE path='".$pathname."/".$nodeFile."');

3) The following query is used to select the raw data from the CaptureFlows table that is associated with the node of interest.

   mysqli_query($connection, "SELECT * FROM CaptureFlows WHERE path='".$pathname."/".$nodeFile."' AND time = '".$time."');

4) The following query is used to select the raw data from the MgenCaptureFlows table that is associated with the node of interest.

   mysqli_query($connection, "SELECT * FROM MgenCaptureFlows WHERE path='".$pathname."/".$nodeFile."' AND time = '".$time."');

5) The following query is used to select the raw data from the Routes table that is associated with the node of interest.

   mysqli_query($connection, "SELECT * FROM Routes WHERE path='".$pathname."/".$nodeFile."' AND time = '".$time."');

7.3.4 ScenarioResults

The following query is used to select the data from the Arff table that is associated with a scenario, where a scenario is distinguished by the pathname.

mysqli_query($connection, "SELECT * FROM Arff WHERE path='".$pathname."');

7.3.5 runConflictDetector

The following curl statements are used to request the RunConflictDetector REST service.

```
curl_init("http://localhost:8080/RestWebserviceExample/sedap/conflictdetector/cd");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_exec($ch);
```

7.3.6 runNewScenario

The following curl statements are used to request the NewScenario REST service. The attributes for the new scenario, selected by the user, are passed as parameters through the URL that is in the curl_init() statement.

```
curl_init("http://localhost:8080/RestWebserviceExample/sedap/newscenar
io/$topology,$protocol,$attack,$attackNode");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_exec($ch);
```

8. Use Case

The SEDAP web interface has been tested with, and is compatible with, the following browsers: Internet Explorer, Google Chrome, and Iceweasel. The interface can be accessed from the localhost server on the Virtual Machine using the following URL: localhost/Pages

Currently, user profiles are not supported. A user is directed to the home screen by default and has access to every feature of the SEDAP interface. The following describes the use of the SEDAP frontend to analyze a sample dataset.

8.1 Executing a New Scenario

1) Initially, the navigation bar displays at the top-left corner of the application (see Fig. 10). Click the **New Scenario** tab.

Fig. 10 SEDAP navigation pane New Scenario tab

2) A page titled **New Scenario** displays on the screen with 5 drop-down menus (see Fig. 11). Each drop-down menu includes the options for the attributes required to run a new scenario, where the attributes are the names to the left of each drop-down menu. The options shown in Fig. 11 are selected by default.

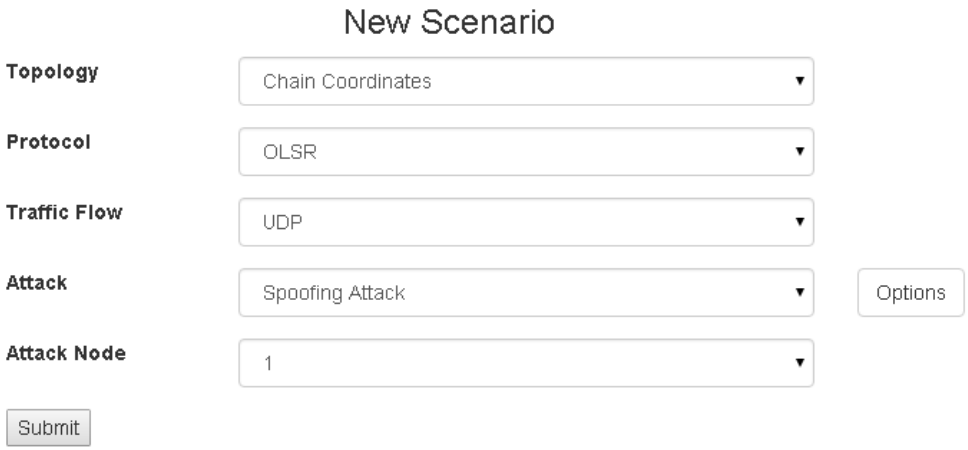

Fig. 11 SEDAP New Scenario page

In the case that the **Spoofing Attack** is selected as the **Attack Attribute**, the **Options** button to the right of the attribute is enabled. For all other options, the **Options** button is disabled. Currently, the **Traffic Flow** field selection has no functionality associated with it. CORE traverses through the traffic flows by default.

If the **Spoofing Attack** option is selected for the **Attack Attribute**, the following steps must be performed to select the configuration options, otherwise, skip to Step 6.

3) Click the **Options** button.

4) A modal with the title **Configuration Options** displays. Select **Node 5** from the **Vulnerable Node** drop-down menu shown in Fig. 12.

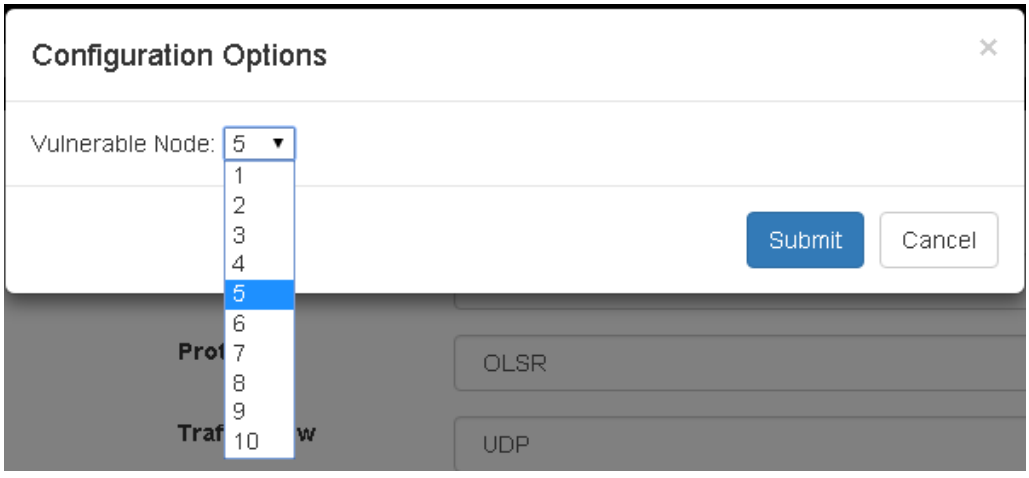

Fig. 12 SEDAP victim node selection

5) Click **Submit**.

6) Click **Submit** in the **New Scenario** page.

7) The pop-up alert shown in Fig. 13 displays. Click **OK**.

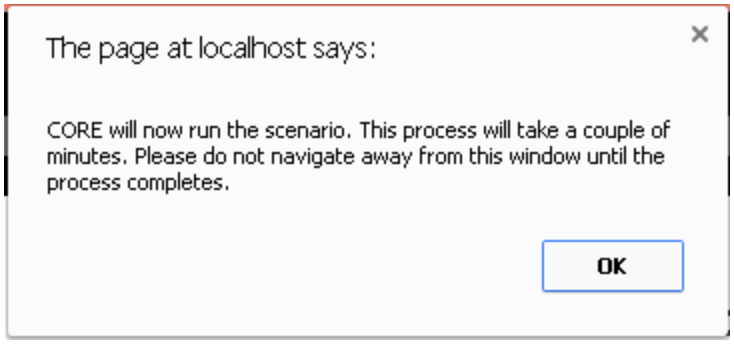

Fig. 13 CORE executive indicator

CORE runs the new scenario and the results stored. Do not navigate from the **New Scenario** page until it has completed.

8) After CORE runs the new scenario and the results are stored, the pop-up alert shown in Fig. 14 displays. Click **OK**.

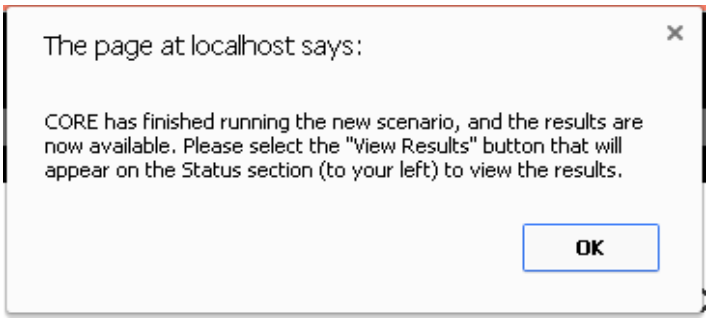

Fig. 14 CORE executive completed indicator

9) The new scenario is listed on the **Status** section as shown in Fig. 15. Click the **View Results** button to view the results. This redirects you to the **Results** page.

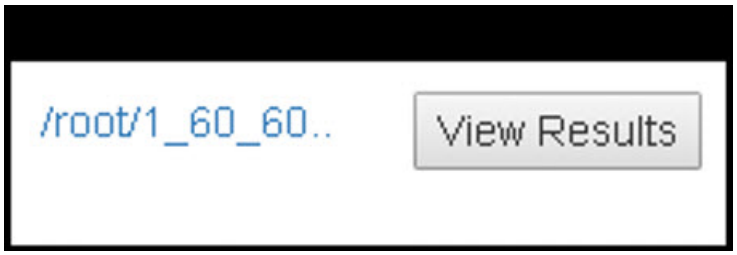

Fig. 15 SEDAP View Results pane

8.2 View Results and Raw Data

This section includes step-by-step instructions for viewing the results.

1) Navigate to the **Results** page. The **Results** page displays the results for the scenario of interest shown in Fig. 16.

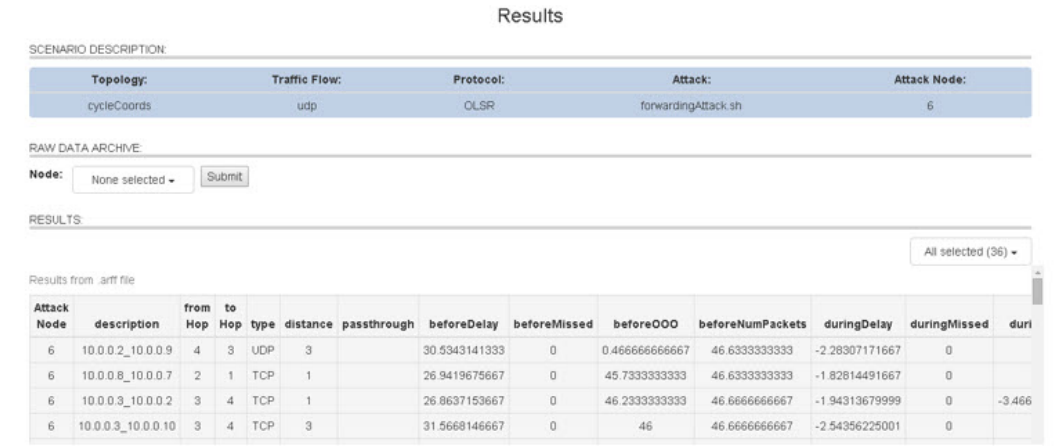

Fig. 16 SEDAP Results page

2) Click the button labeled **All selected (36)** that is located above the results table (see Fig. 17).

3) Deselect the **description** option. Notice the column labeled **description** is removed from the table, allowing you to filter attributes.

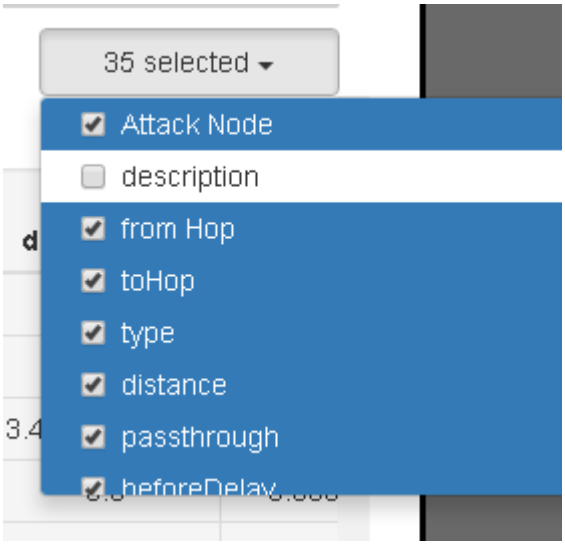

Fig. 17 SEDAP results field filter

4) In the **Raw Data Archive** section, select **6** from the drop-down list labeled **Node** shown in Fig. 18.

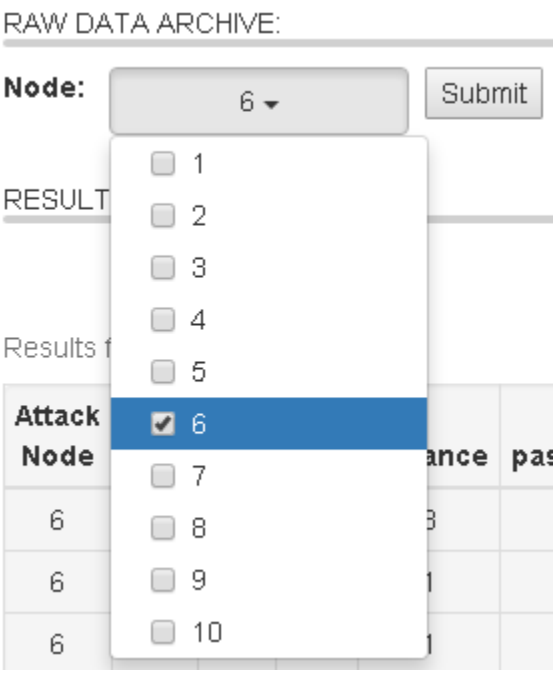

Fig. 18 SEDAP raw node data selection

5) Click **Submit**. This opens a new window that displays the raw data. This process takes a few minutes. During this time DO NOT navigate away from the **Results** page. You must wait until the new window populates the raw data table entirely.

6) A new window displays with the raw data for the node that was selected in Step 6 shown in Fig. 19.

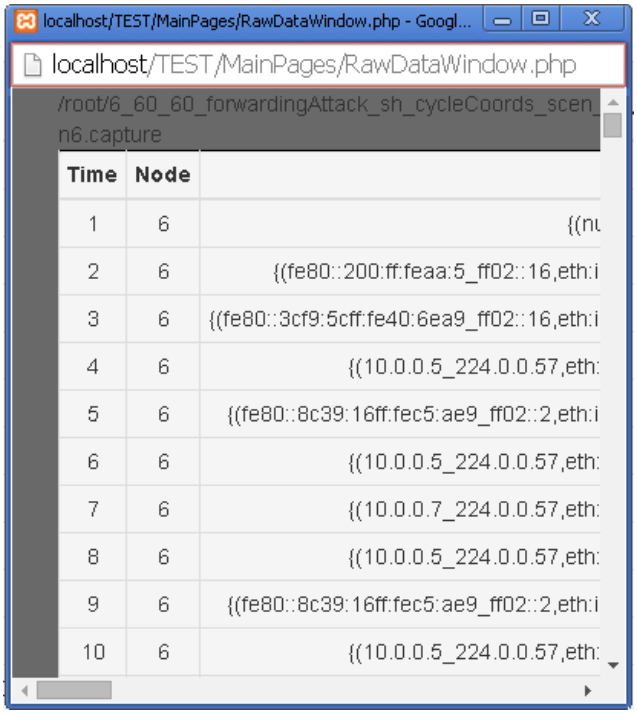

Fig. 19 SEDAP raw data window

The pathname for the scenario that the node is associated with is displayed above the table. The filename for the node is also displayed. If the node is the attack node, it is followed by the .capture extension; otherwise, it is followed by the .mgencapture extension.

7) Close the window.

8) To return to the main **SEDAP** page, click the **SEDAP** tab (located at the top-left corner of the page).

8.3 Run Conflict Detector

This section includes step-by-step instructions for running the Conflict Detector.

1) Click the **Conflicts** tab.

2) On the **Conflict Detection** page (see Fig. 20), Click the **Re-run Conflict Detector** button.

Fig. 20 SEDAP Conflict Detection page

3) The pop-up alert shown in Fig. 21 displays. Click **OK**.

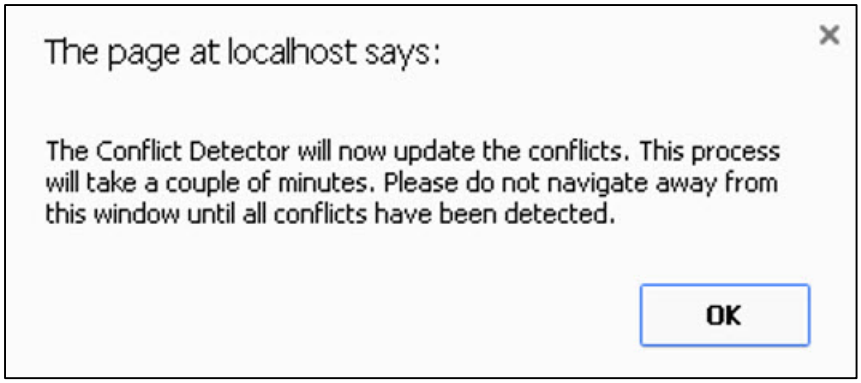

Fig. 21 SEDAP conflict detection execution indicator

The Conflict Detector starts executing and conflicts are updated. Do not navigate from the **Conflict Detection** page until it has completed.

4) After the conflicts are updated, the pop-up alert shown in Fig. 22 displays. Click **OK**.

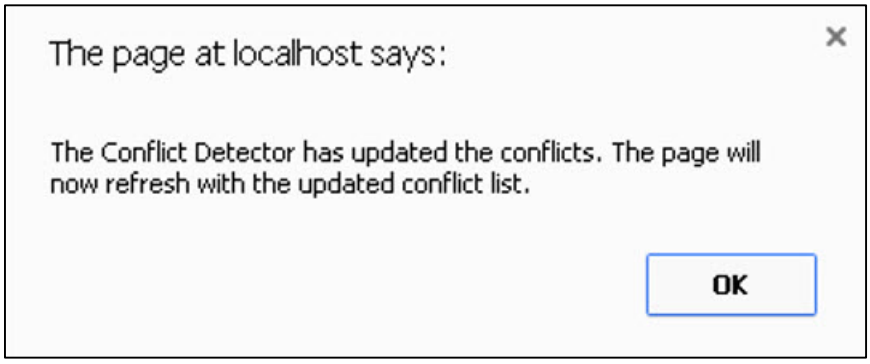

Fig. 22 SEDAP conflict detection completed indicator

5) The updated conflicts display on the screen.

6) Click the **SEDAP** tab to move to the home screen.

8.4 View Conflicts

This section includes step-by-step instructions for viewing conflicts. A conflict has occurred when 2 or more rows in the database share the same key, but have a different value. The table encapsulated within the light-blue container shows the key for the rows that are in conflict. The conflict count to the right shows how many times the value for those rows is true or false.

1) Move to the navigation bar (at the top-left corner of the application) and Click the **Conflicts** tab.

2) The conflicts display on the screen. Click the triangle (▶) to the left of the data entry DuringLinkLost=TRUE.

3) An expanded tree displays shown in Fig. 23. Double-click the first row under the data entry, DuringLinkLost=TRUE, to view the results for that row. This redirects you to the **Results** page.

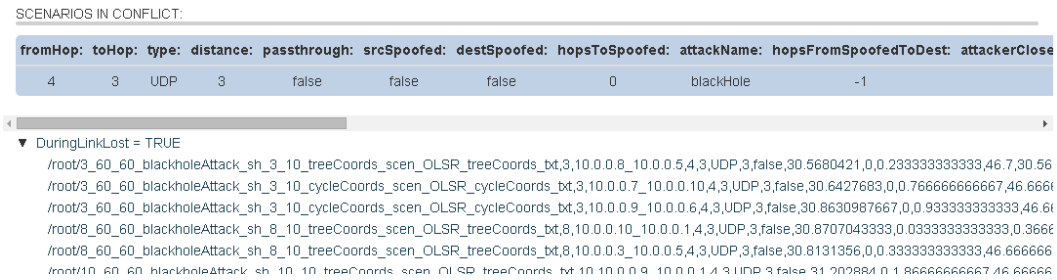

Fig. 23 SEDAP detailed conflict drop-down view

9. Conclusions

We have described the first step in building an analysis platform for MANET scenario log files. With this system, analysts are now able to calculate potential impacts of certain network-layer attacks. This system will eventually be expanded to support various emulators and simulators. This provides an integrated environment for conducting analysis of attack impacts on MANETs. Additionally, we plan to enhance the capability of SEDAP by providing a mechanism (through a plugin software architecture) for conducting model quality comparison, including verification and validation. This consists of using emulators to collect data (running real binaries in realistic environments) and then comparing against simulation outputs.

10. References

Official Navy Website. Networks and Communication Systems Branch. Common Open Research Emulator (CORE). [accessed 2015 Sep 29]. http://www.nrl.navy.mil/itd/ncs/products/core.

List of Symbols, Abbreviations, and Acronyms

Arff	Attribute-Relation File Format
ARL	US Army Research Laboratory
CEPD	Cybersecurity and Electromagnetic Protection Division
CORE	common open research emulator
ISO	International Organization of Standardization
MANE	mobile ad hoc network emulator
MANET	mobile ad hoc network
NRL OLSR	US Naval Research Laboratory Optimized Link-State Routing
SEDAP	scenario execution data analysis platform
SLAD	Survivability/Lethality Analysis Directorate
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WEKA	Waikato Environment for Knowledge Analysis

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RSRCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (WORD VERSION)	US ARMY RSRCH LAB ATTN RDRL SLE MARISSA WITHERS BLDG 1624 RM 210 WSMR NM 88002-5513
2 (PDF)	DIR USARL RDRL SLE I J C ACOSTA J JACQUEZ

INTENTIONALLY LEFT BLANK.